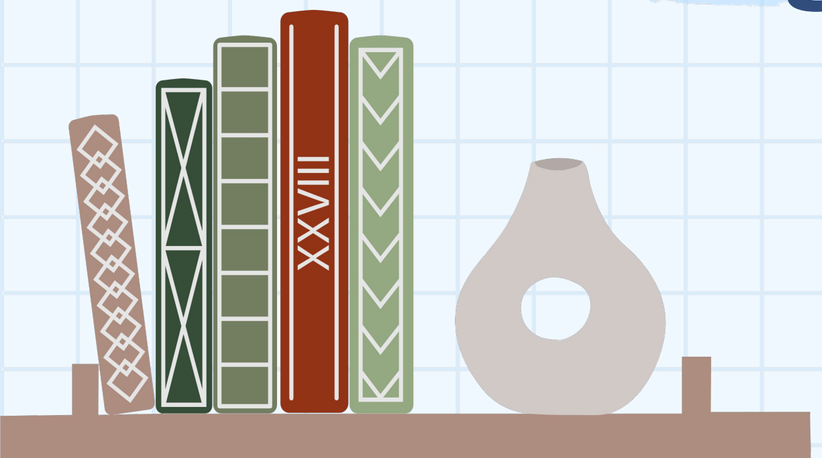
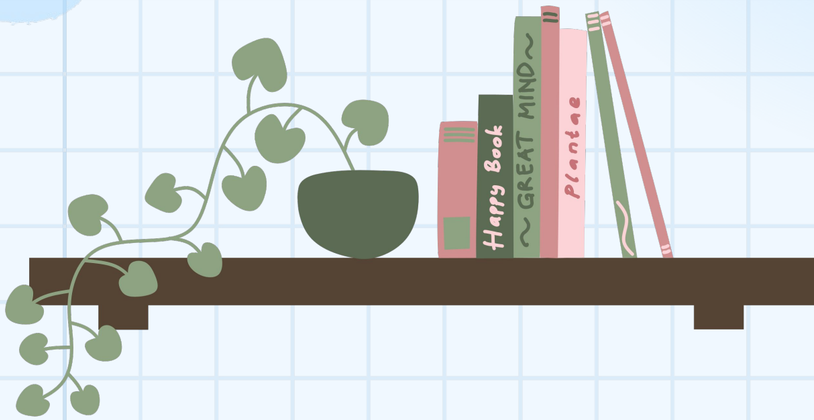
BS./BSC.

Applied AI and Data Science

Algorithmic Thinking & its Applications





Recall: Modules

- Modules provide extra functions, variables
  - **Example**: math provides math.cos(), math.pi
  - Access them with the import command
- Python provides a lot of them for us
- **This Lecture**: How to make modules
  - Pulsar to *make* a module
  - Python to *use* the module

Two different programs

We Write Programs to Do Things

- Functions are the **key doers**

Function Call Function Definition

- Command to **do** the function

```
>>>
plus(23)
24
>>>
```

- Defines what function **does**

```
def plus(n):
    return
    n+1
```

- **Parameter**: variable that is listed within the parentheses of a method header.
- **Argument**: a value to assign to the method parameter when it is called

We Write Programs to Do Things

- Functions are the **key doers**

Function Call Function Definition

- Command to **do** the function
- Defines what function **does**

```
>>>
plus(23)
24
>>>
```

Function **Header**

```
def plus(n):
    return
    n+1
```

- **Parameter**: variable that is listed within the parentheses of a method header.
- **Argument**: a value to assign to the method parameter when it is called

We Write Programs to Do Things

- Functions are the **key doers**

Function Call Function Definition

- Command to **do** the function

```
>>>
plus(23)
24
>>>
```

- Defines what function **does**

```
def plus(n):
    return
    n+1
```

Function **Header**

Function **Body** (indented)

- **Parameter**: variable that is listed within the parentheses of a method header.
- **Argument**: a value to assign to the method parameter when it is called

We Write Programs to Do Things

- Functions are the **key doers**

Function Call Function Definition

- Command to **do** the function
- Defines what function **does**

```
>>>
plus(23)
24
```

argument to assign to n

Function **Header**

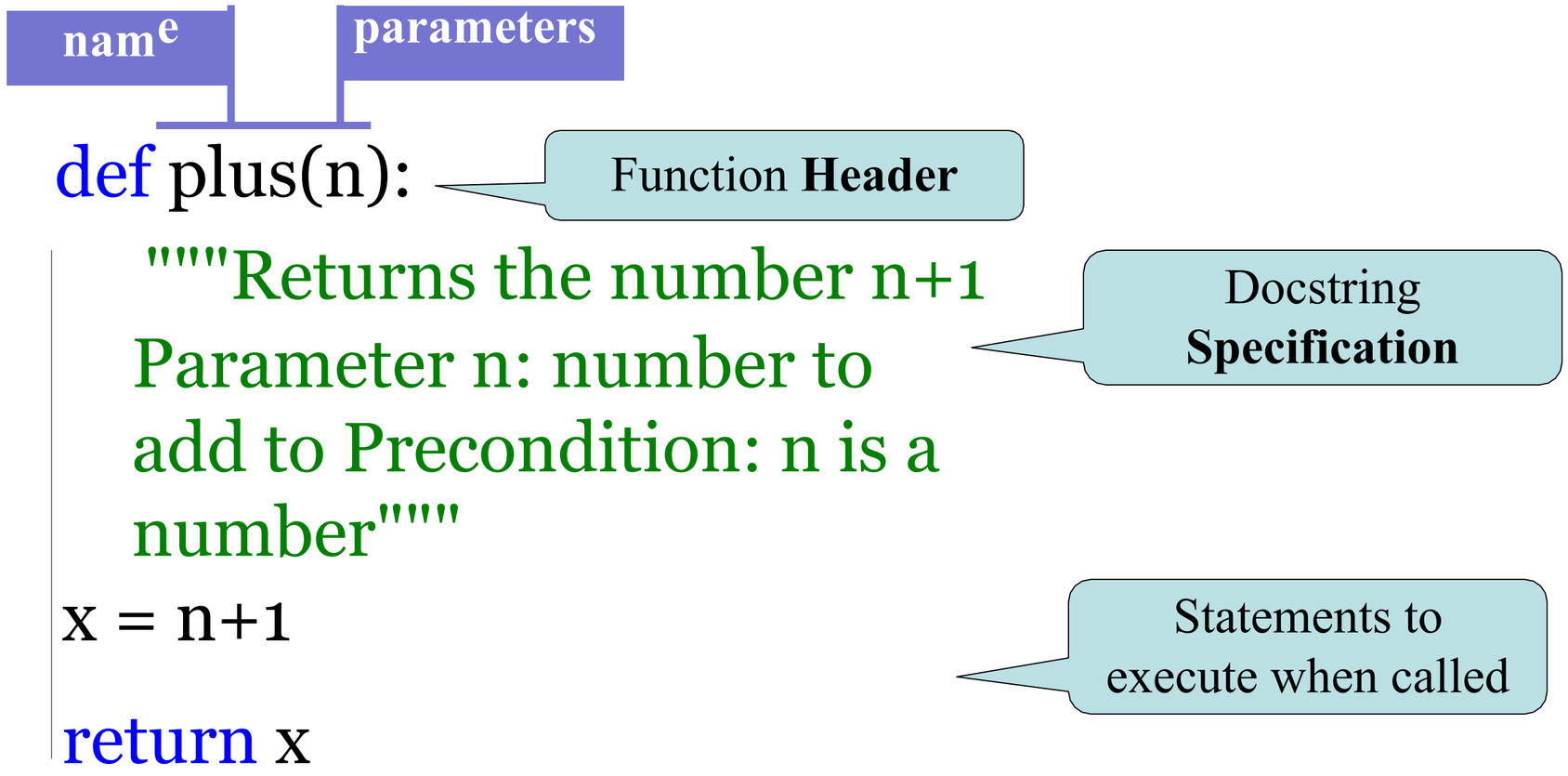
```
def plus(n):
    return
```

declaration of **parameter** n

Function **Body** (indented)

- **Parameter**: variable that is listed within the parentheses of a method header.
- **Argument**: a value to assign to the method parameter when it is called

Anatomy of a Function Definition



The return Statement

- **Format**: return <*expression*>
  - Used to evaluate ***function call*** (as an expression)
  - Also stops executing the function!
  - Any statements after a **return** are ignored
- **Example**: temperature converter function

```
def to_centigrade(x):
    """Returns: x converted to
    centigrade""" return 5*(x-32)/9.0
```

A More Complex Example

Function Definition Function Call

```
def foo(a,b):
    """Return something
       Param a: number
       Param b:
       number"""
    x = a
    y = b
    return x*y+y
```

```
>>> x = 2
>>> foo(3,4)
```

x ?

What is in the box?

A More Complex Example

Function Definition Function Call

```
def foo(a,b):
    """Return something
        Param a: number
        Param b: number"""
    x = a
    y = b
    return x*y+y
```

```
>>> x = 2
>>> foo(3,4)
```

x ?

What is in the box?

A: 2
B: 3
C: 16
D: Nothing!
E: I do not know

A More Complex Example

Function Definition Function Call

```
def foo(a,b):
    """Return something
       Param a: number
       Param b: number"""
    x = a
    y = b

    return x*y+y
```

```
>>> x = 2
>>> foo(3,4)
```

x ?

What is in the box?

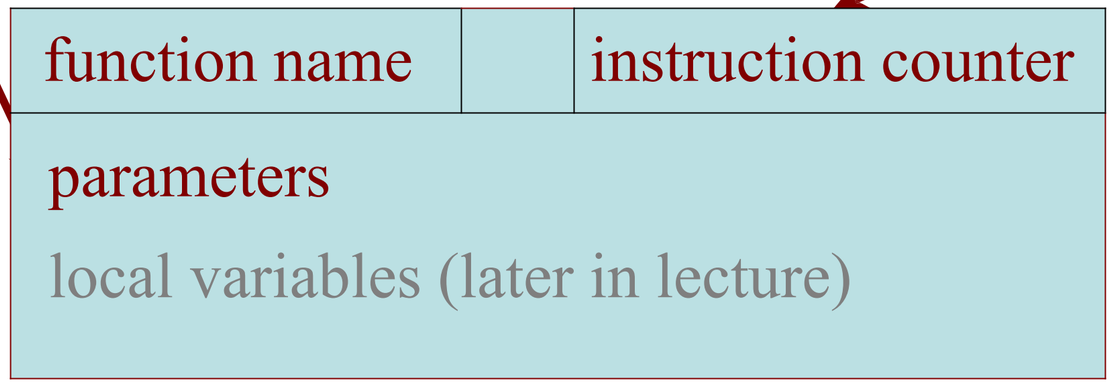
A: 2
B: 3
C: 16 **CORRECT**
D: Nothing!
E: I do not know

Understanding How Functions Work

- **Function Frame**: Representation of function call
- A **conceptual model** of Python

Draw parameters as variables (named boxes)

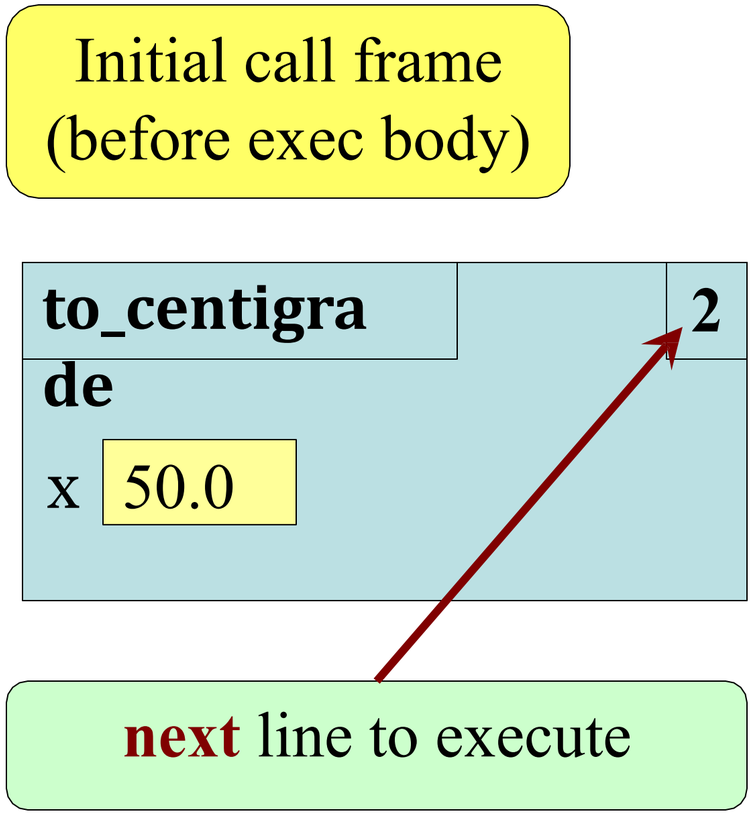
- Number of statement in the function body to execute next
- **Starts with line no. of body**



Example: to_centigrade(50.0)

1. Draw a frame for the call
2. Assign the argument value to the parameter (in frame)
3. Execute the function body
   - Look for variables in the frame
   - If not there, look for global variables with that name
4. Erase the frame for the call

```
1  def
2      to_centigrade(x):
       return
       5*(x-32)/9.0
```

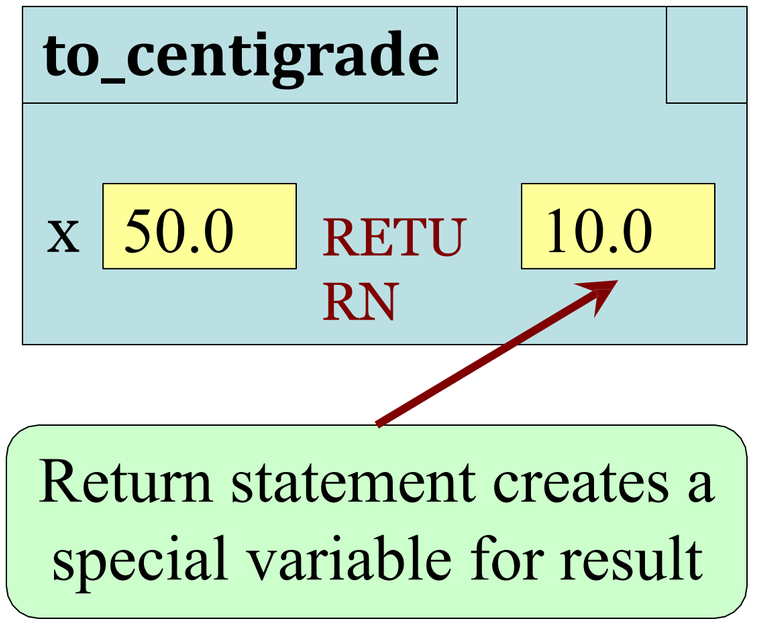


Example: to_centigrade(50.0)

1. Draw a frame for the call
2. Assign the argument value to the parameter (in frame)
3. Execute the function body
   - Look for variables in the frame
   - If not there, look for global variables with that name
4. Erase the frame for the call

```
1  def to_centigrade(x):
2      return 5*(x-32)/9.0
```

Executing the return statement

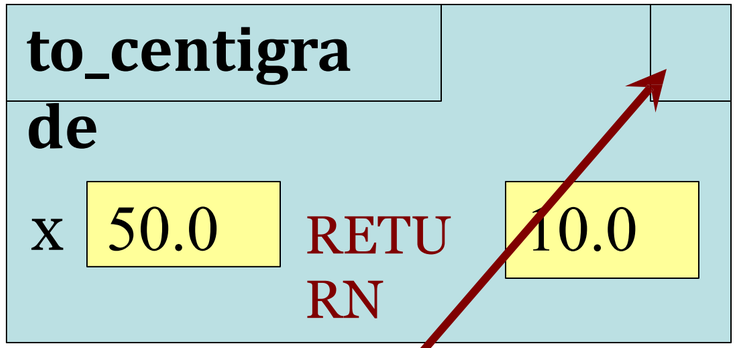


Example: to_centigrade(50.0)

1. Draw a frame for the call
2. Assign the argument value to the parameter (in frame)
3. Execute the function body
   - Look for variables in the frame
   - If not there, look for global variables with that name
4. Erase the frame for the call

```
1  def to_centigrade(x):
2      return 5*(x-32)/9.0
```

Executing the return statement



The return terminates; no next line to execute

Example: to_centigrade(50.0)

1. Draw a frame for the call
2. Assign the argument value to the parameter (in frame)
3. Execute the function body
   - Look for variables in the frame
   - If not there, look for global variables with that name
4. Erase the frame for the call

```
1  def to_centigrade(x):
2      return 5*(x-32)/9.0
```

ERASE WHOLE FRAME

But don't actually erase on an exam

Call Frames vs. Global Variables

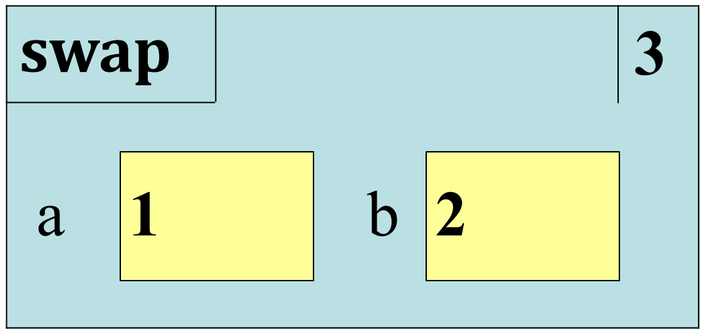
The specification is a **lie**:

```
1  def swap(a,b):
2      """Swap global a &
3      b""" tmp = a
4      a = b
5      b = tmp
```

```
>>> a = 1
>>> b = 2
>>>
swap(a,b)
```

Global Variables

Call Frame



Call Frames vs. Global Variables

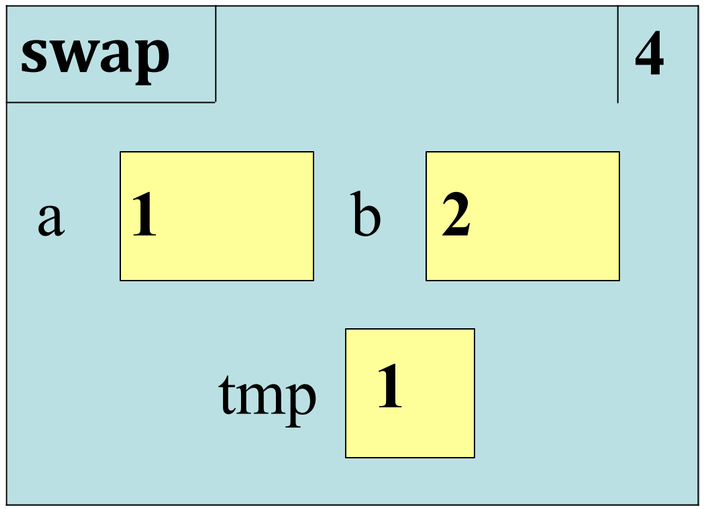
The specification is a **lie**:

```
1  def swap(a,b):
2      """Swap global a &
3      b""" tmp = a
4      a = b
5      b = tmp
```

```
>>> a = 1
>>> b = 2
>>>
swap(a,b)
```

Global Variables

Call Frame



Call Frames vs. Global Variables

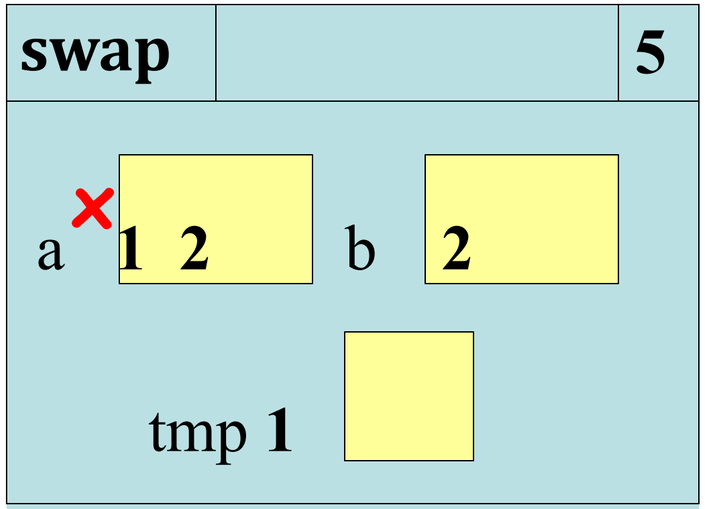
The specification is a **lie**:

```
1  def swap(a,b):
2      """Swap global a &
3      b""" tmp = a
4      a = b
5      b = tmp
```

```
>>> a = 1
>>> b = 2
>>>
swap(a,b)
```

Global Variables

Call Frame



Call Frames vs. Global Variables

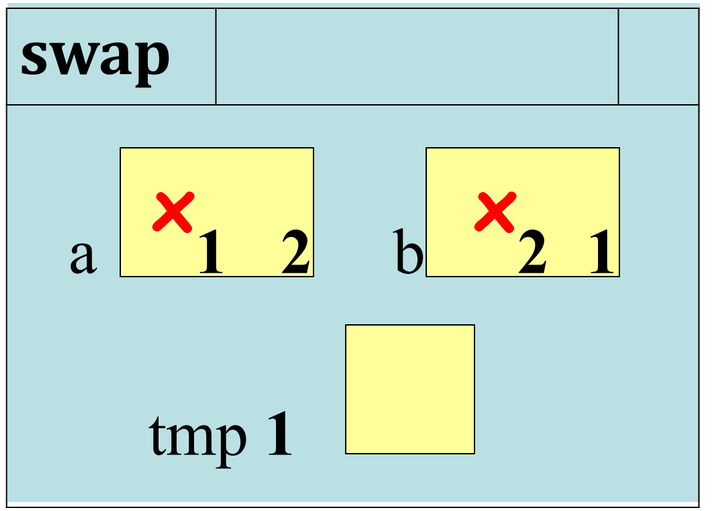
The specification is a **lie**:

```
1  def swap(a,b):
2      """Swap global a &
3      b""" tmp = a
4      a = b
5      b = tmp
```

```
>>> a = 1
>>> b = 2
>>>
swap(a,b)
```

Global Variables

Call Frame



Call Frames vs. Global Variables

The specification is a **lie**:

```
1  def swap(a,b):
2      """Swap global a &
3      b""" tmp = a
4      a = b
5      b = tmp
```

```
>>> a = 1
>>> b = 2
>>>
swap(a,b)
```

Global Variables

a **1** b **2**

Call Frame

ERASE THE FRAME

Exercise Time

Function Definition Function Call

The specification is a **lie**:

```
1  def swap(a,b):
2      """Swap global a & b"""
3      tmp = a
4      a = b
5      b = tmp
```

```
>>> a = 1
>>> b = 2
>>> swap(a,b)
```

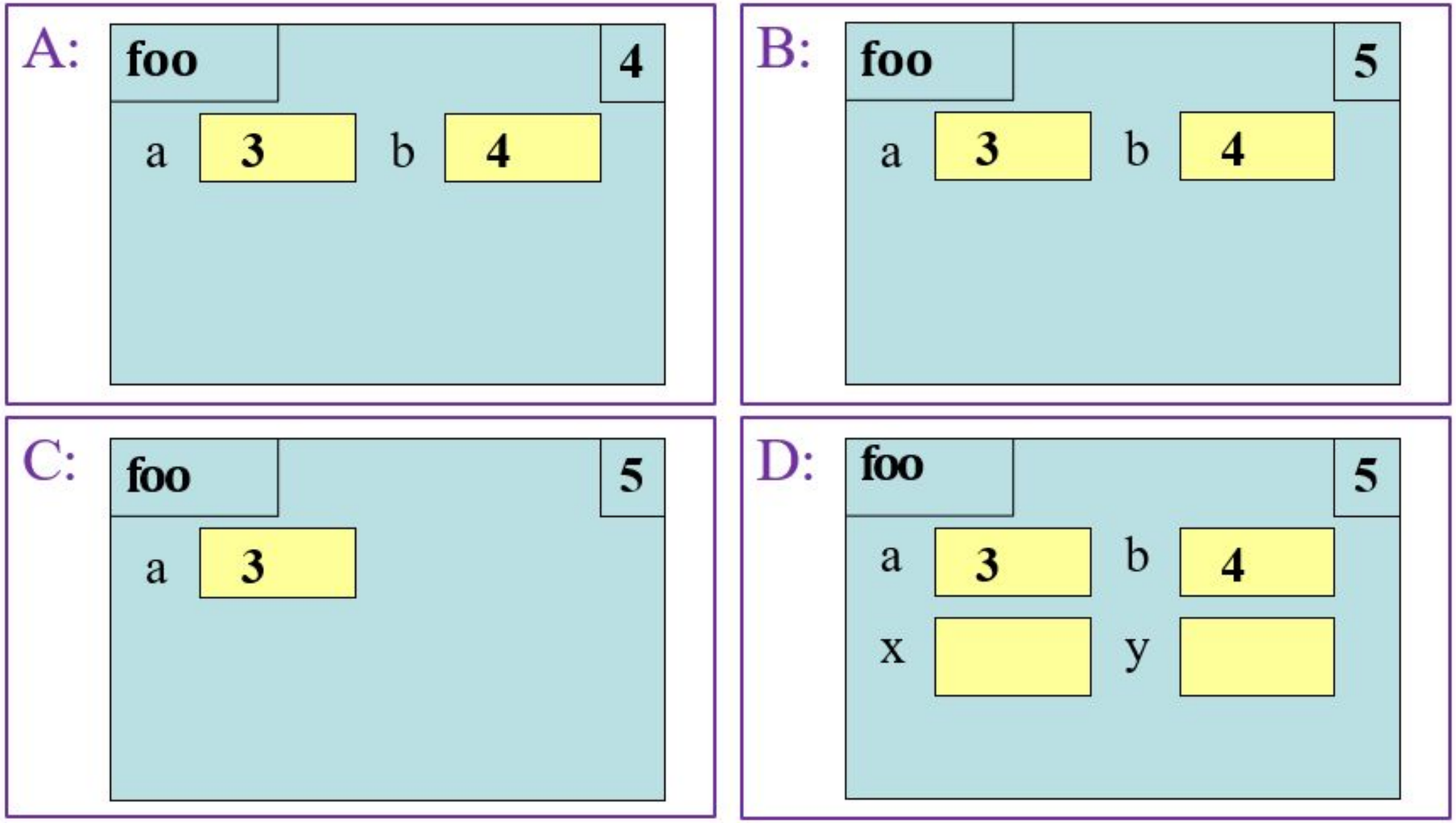
Global Variables

a **1** b **2**

Call Frame

ERASE THE FRAME

Which One is Closest to Your Answer?



Exercise Time

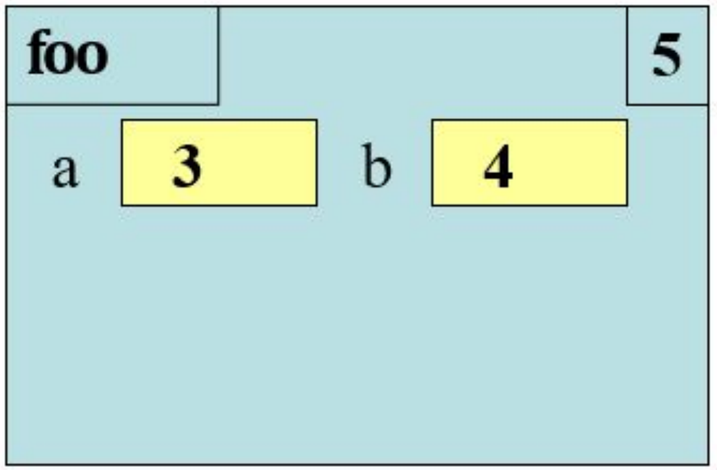
Function Definition

```
1 def foo(a,b):
2     """Return something
3         Param x: a number
4         Param y: a number"""
5     x = a
6     y = b
7     return x*y+y
```

Function Call

```
>>> x = foo(3,4)
```

B:



Exercise Time

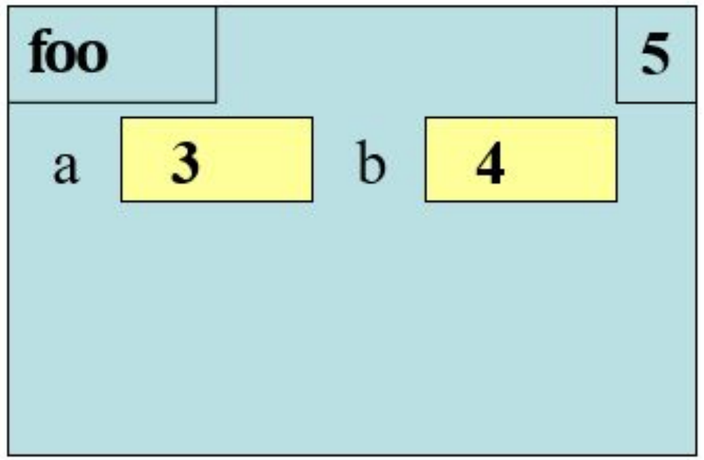
Function Definition

```
1 def foo(a,b):
2     """Return something
3         Param x: a number
4         Param y: a number"""
5     x = a
6     y = b
7     return x*y+y
```

Function Call

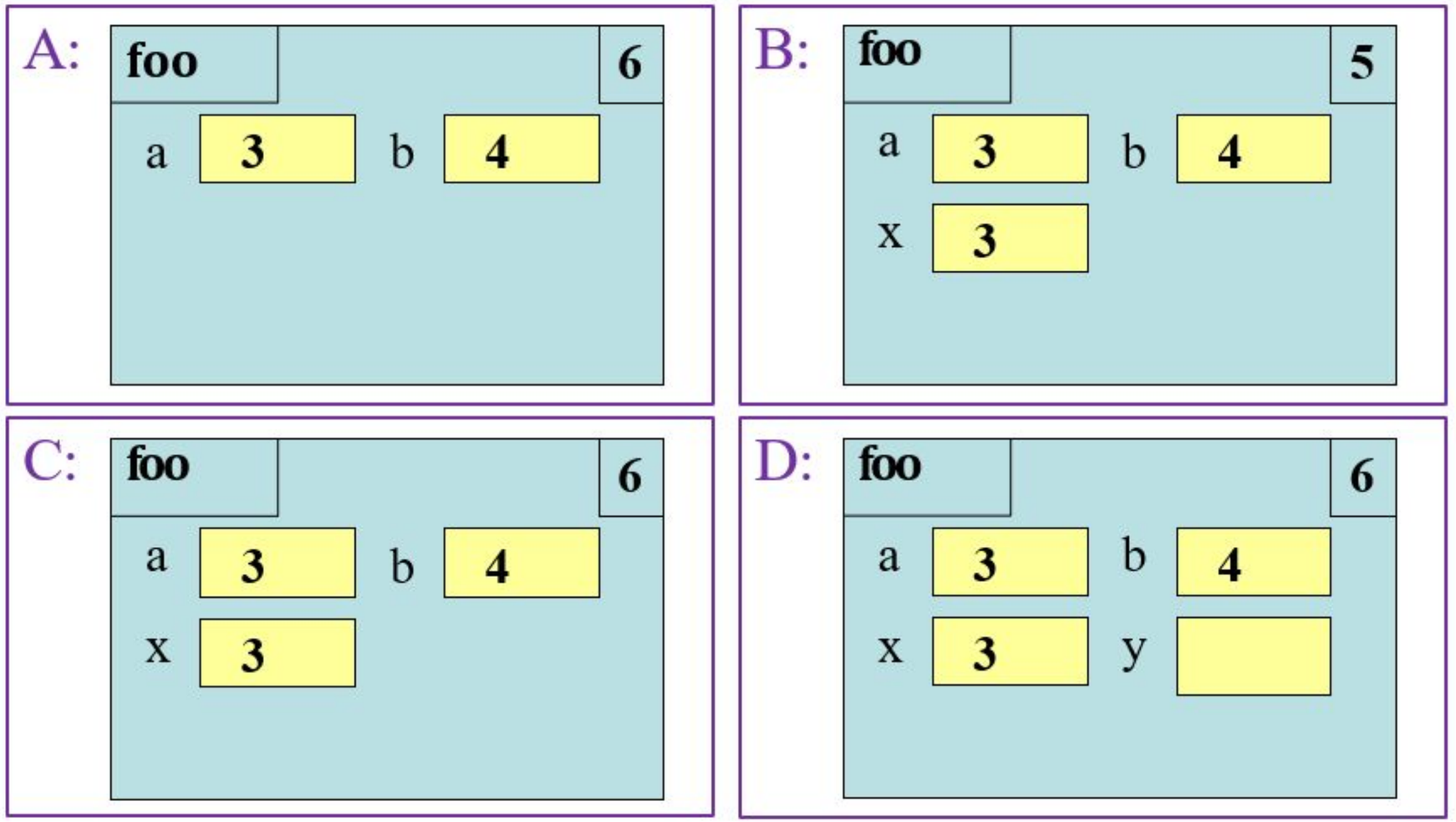
```
>>> x = foo(3,4)
```

B:



What is the **next step**?

Which One is Closest to Your Answer?



Exercise Time

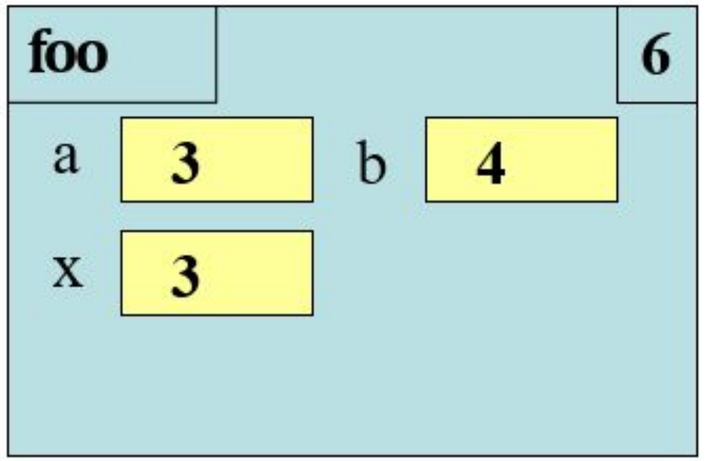
Function Definition

```
1 def foo(a,b):
2     """Return something
3         Param x: a number
4         Param y: a number"""
5     x = a
6     y = b
7     return x*y+y
```

Function Call

```
>>> x = foo(3,4)
```

C:



Exercise Time

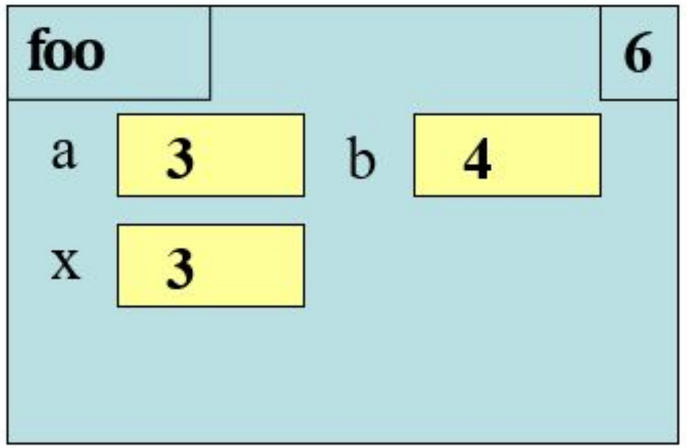
Function Definition

```
1 def foo(a,b):
2     """Return something
3         Param x: a number
4         Param y: a number"""
5     x = a
6     y = b
7     return x*y+y
```

Function Call

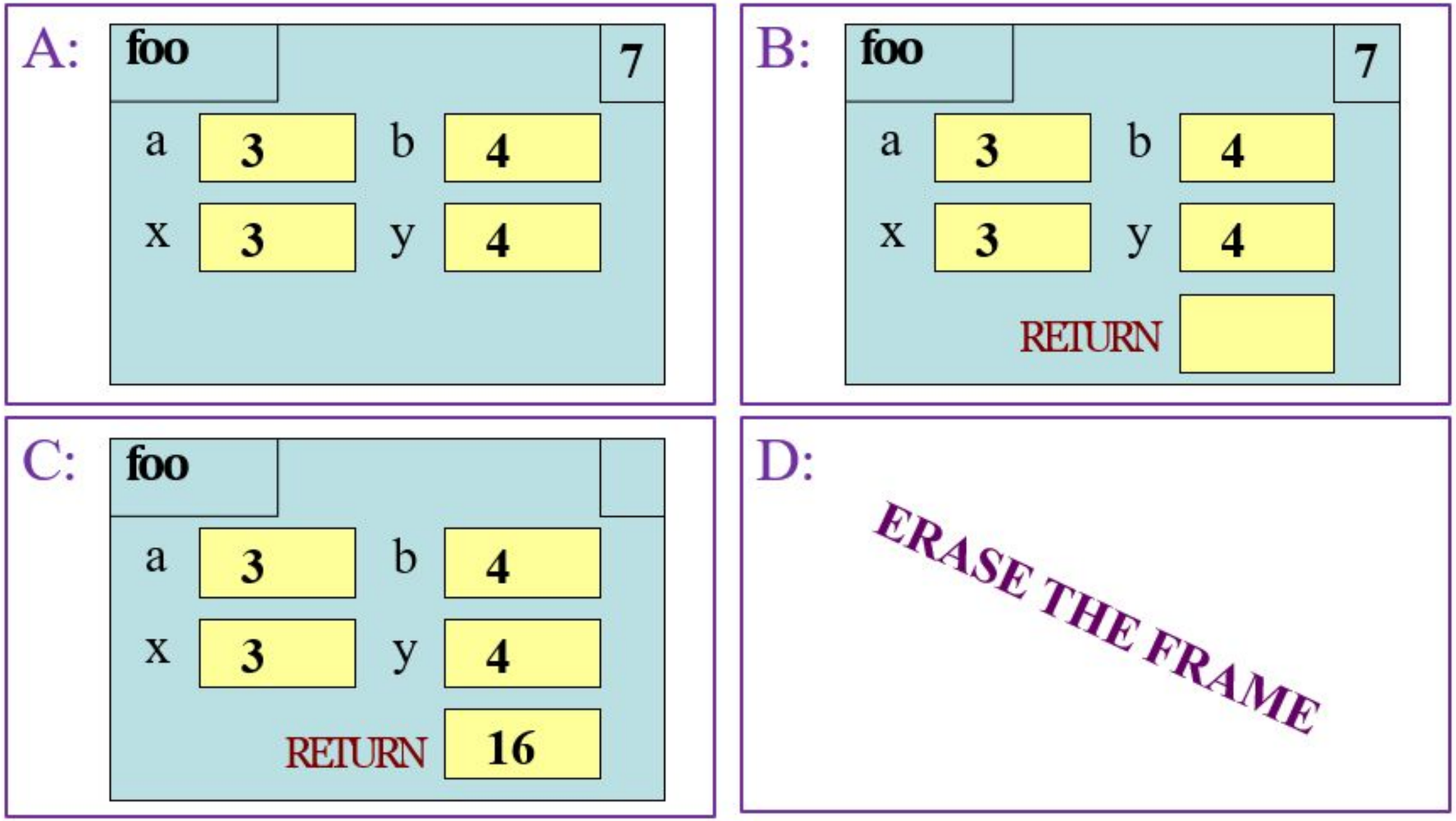
```
>>> x = foo(3,4)
```

C:

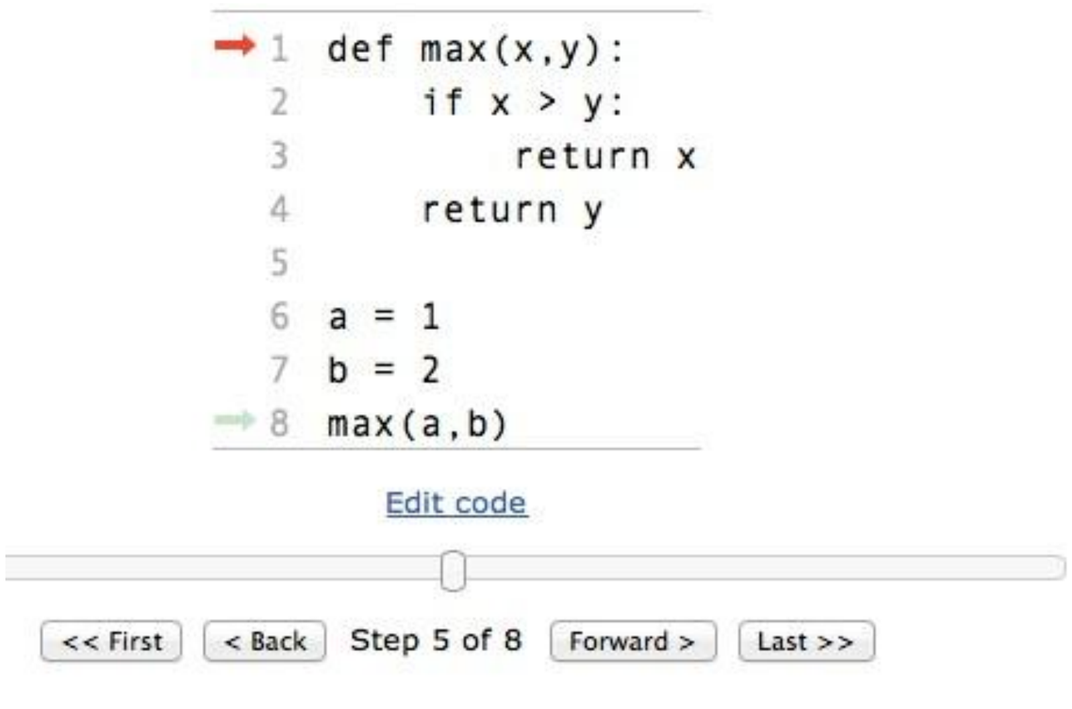


What is the **next step**?

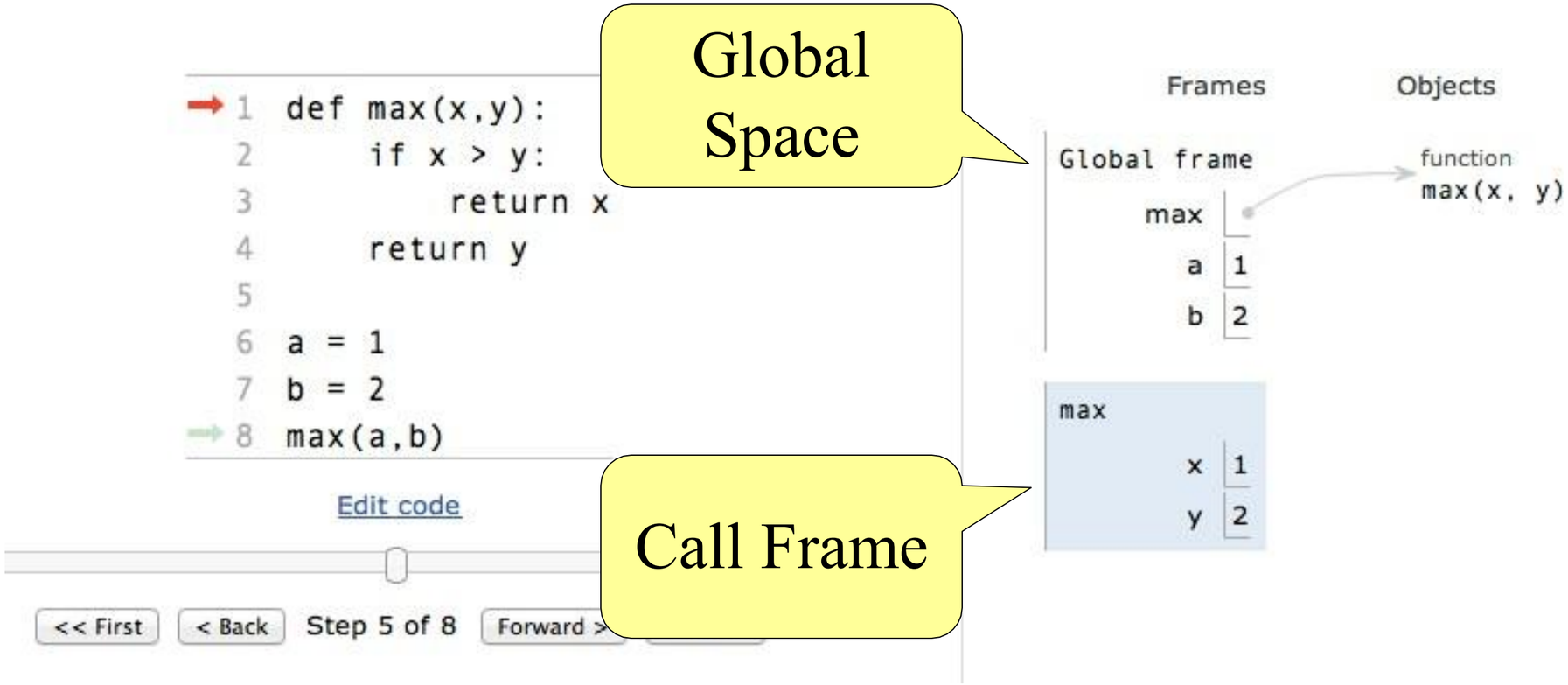
Which One is Closest to Your Answer?



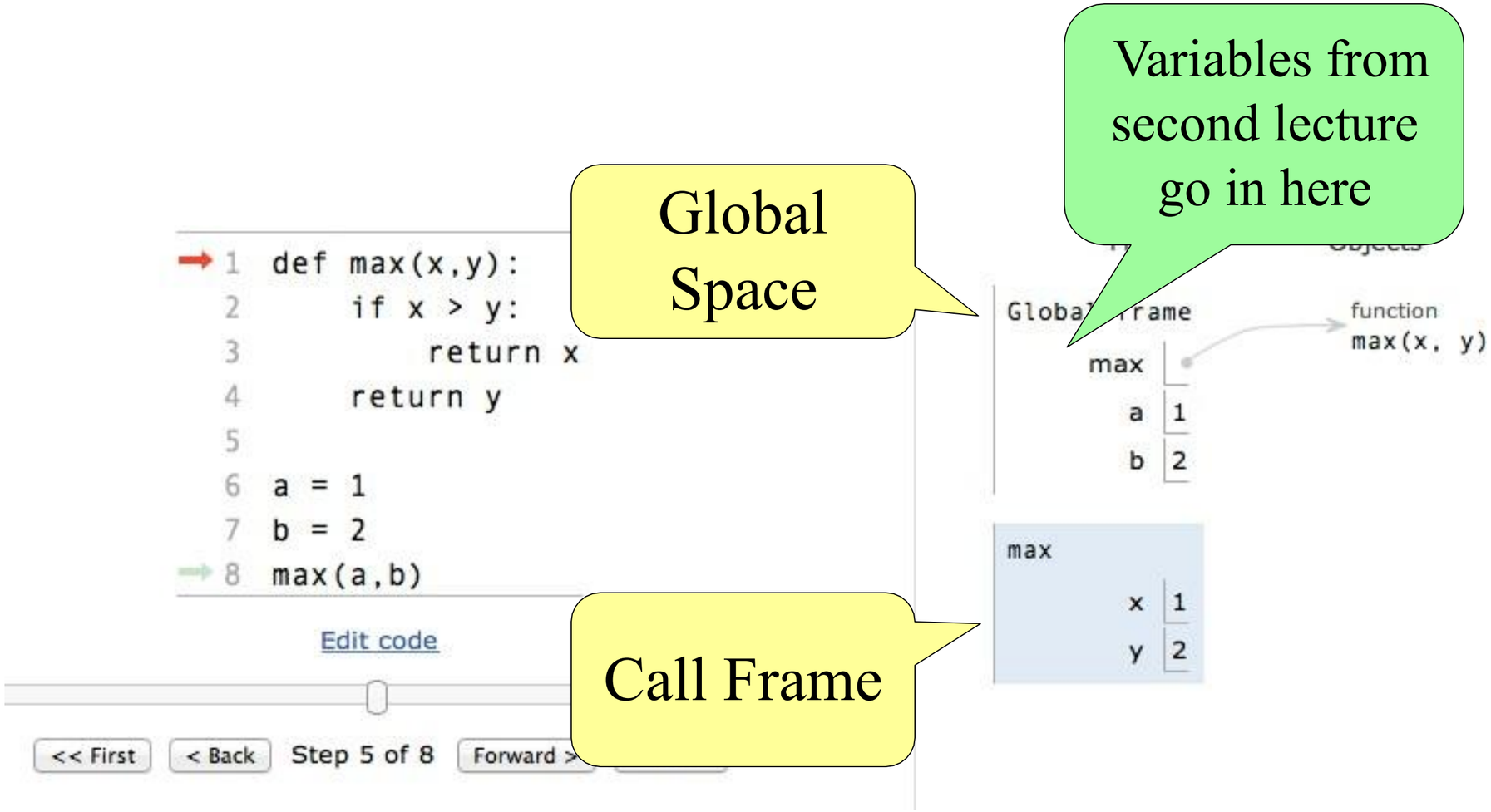
Visualizing Frames: The Python Tutor



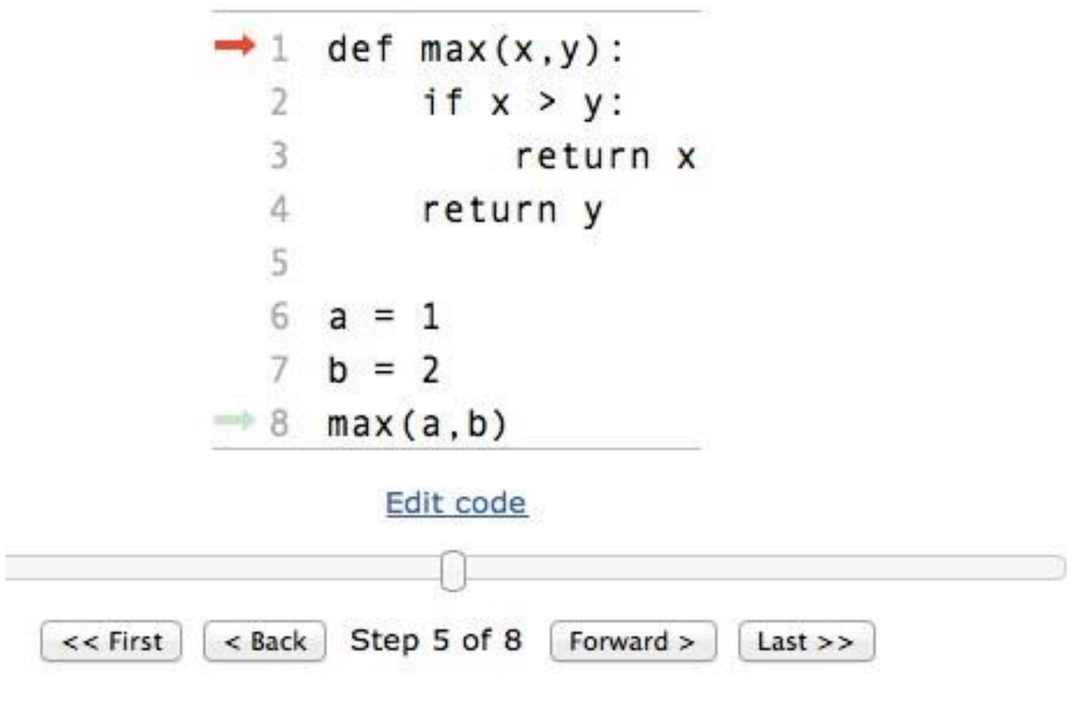
Visualizing Frames: The Python Tutor

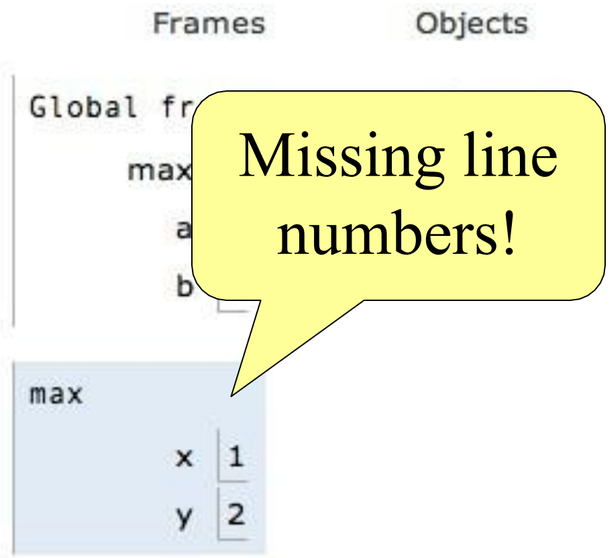


Visualizing Frames: The Python Tutor

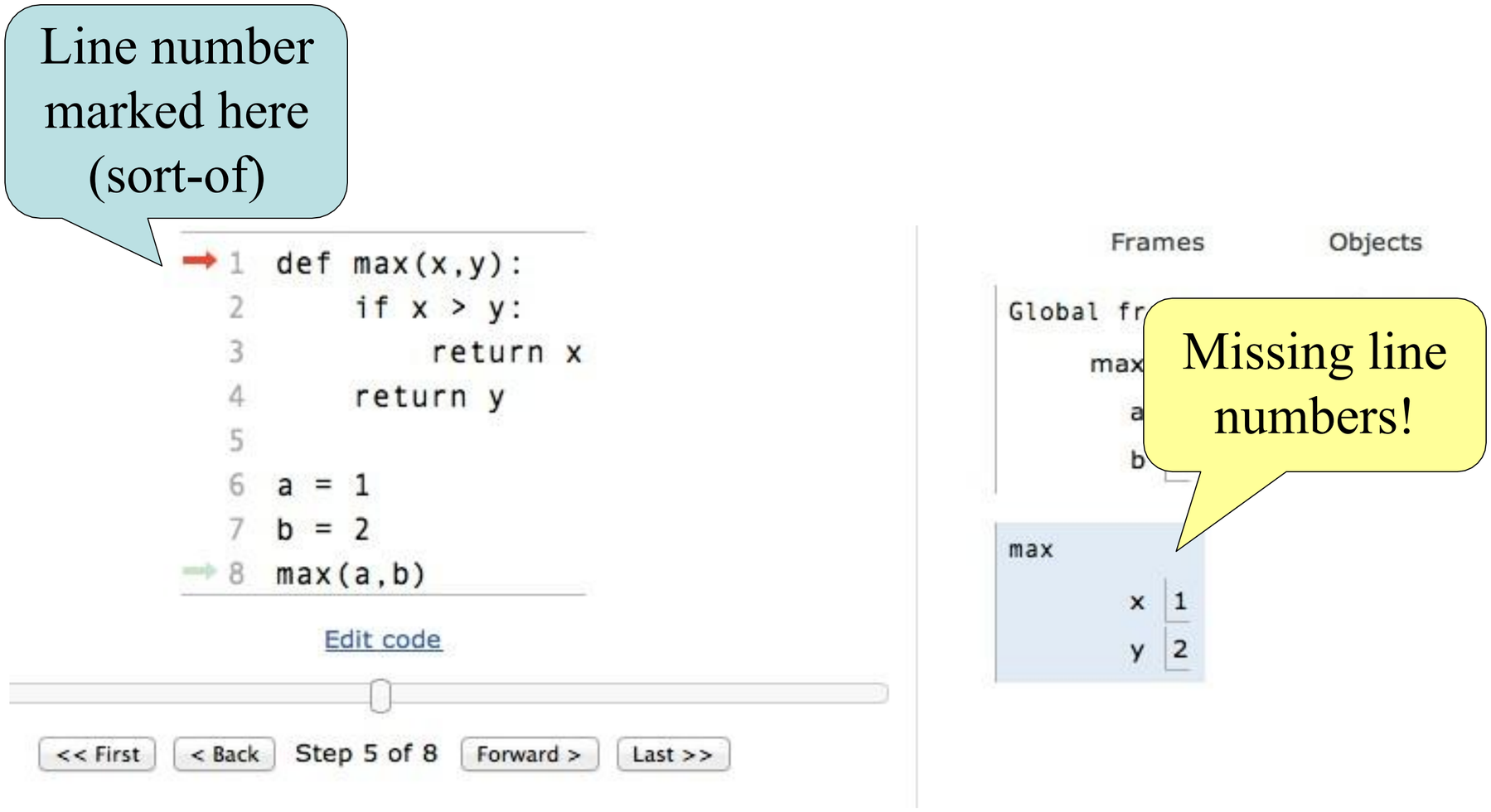


Visualizing Frames: The Python Tutor





Visualizing Frames: The Python Tutor



Thank you